

CURSO DE JAVA CON JDBC

EJERCICIO

MANEJO DE PERSONAS CON JDBC



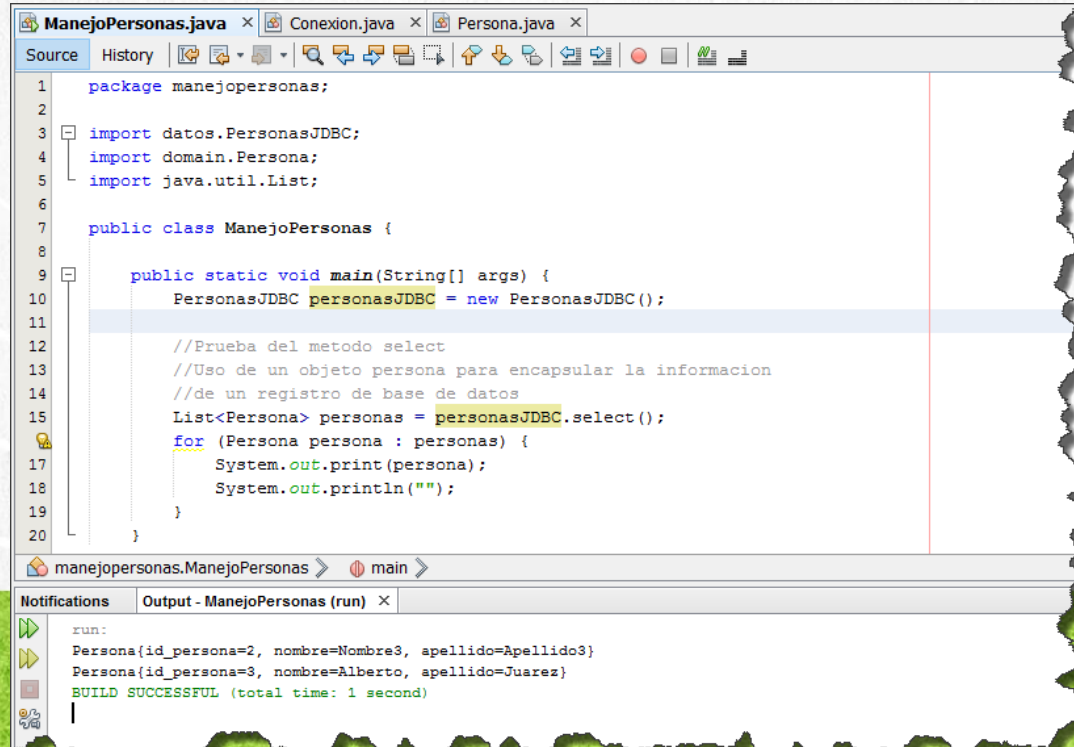
Experiencia y Conocimiento para tu vida

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

Crear un programa para el manejo de objetos personas utilizando JDBC. Al finalizar deberemos observar lo siguiente:



```
1 package manejopersonas;
2
3 import datos.PersonasJDBC;
4 import domain.Persona;
5 import java.util.List;
6
7 public class ManejoPersonas {
8
9     public static void main(String[] args) {
10         PersonasJDBC personasJDBC = new PersonasJDBC();
11
12         //Prueba del metodo select
13         //Uso de un objeto persona para encapsular la informacion
14         //de un registro de base de datos
15         List<Persona> personas = personasJDBC.select();
16         for (Persona persona : personas) {
17             System.out.print(persona);
18             System.out.println("");
19         }
20     }
21 }
```

manejopersonas.ManejoPersonas > main >

Notifications Output - ManejoPersonas (run) X

```
run:
Persona{id_persona=2, nombre=Nombre3, apellido=Apellido3}
Persona{id_persona=3, nombre=Alberto, apellido=Juarez}
BUILD SUCCESSFUL (total time: 1 second)
```

PASO 1. CREACIÓN DEL PROYECTO

Vamos a crear el proyecto ManejoPersonas:

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: ManejoPersonas

Project Location: C:\Cursos\JDBC\Leccion04 Browse...

Project Folder: C:\Cursos\JDBC\Leccion04\ManejoPersonas

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

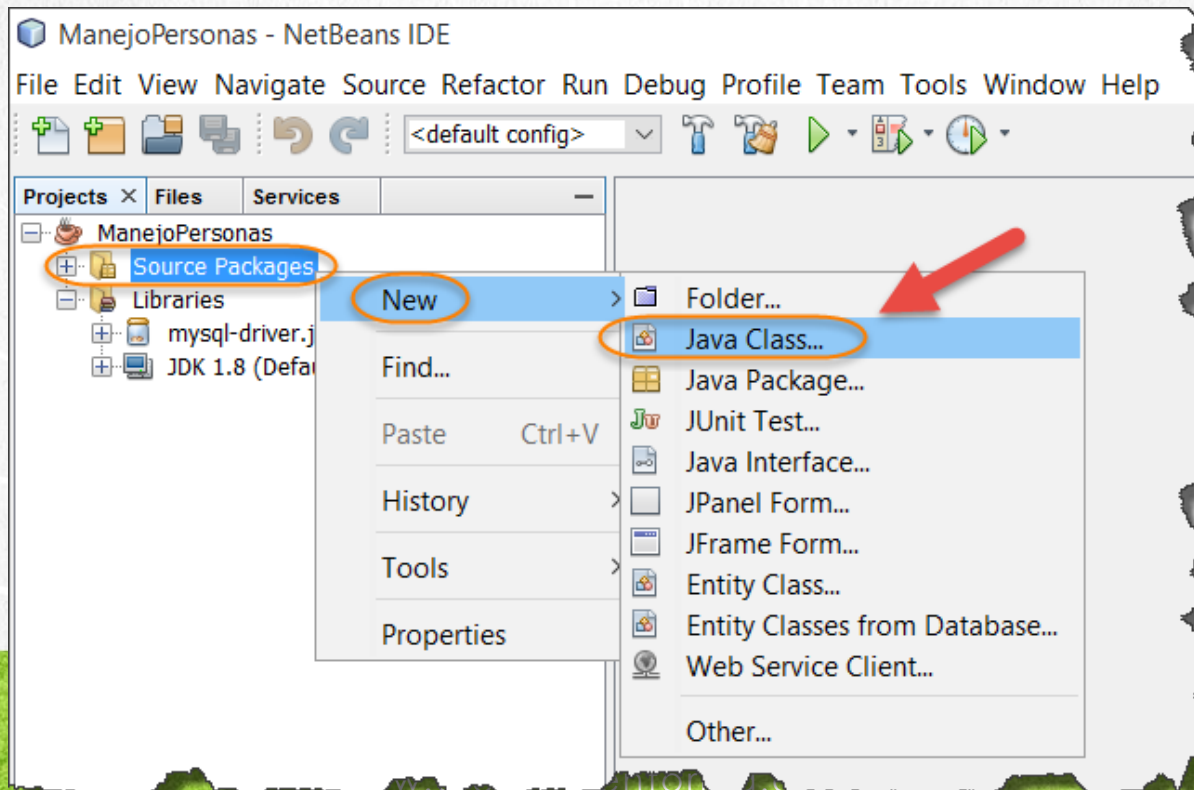
Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class manejopersonas.ManejoPersonas

< Back Next > Finish Cancel Help

PASO 2. CREACIÓN DE UNA CLASE

Vamos a crear una nueva clase:



PASO 2. CREACIÓN DE UNA CLASE (CONT)

Vamos a crear una nueva clase:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

PASO 3. CREACIÓN DE UNA CLASE

Vamos a crear una nueva clase:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

PASO 4. CREACIÓN DE UNA CLASE

Vamos a crear una nueva clase:

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo Persona.java:

Dar click para ir al código

```
package domain;
public class Persona {

    private int id_persona;
    private String nombre;
    private String apellido;

    public int getId_persona() {
        return id_persona;
    }

    public void setId_persona(int idPersona) {
        id_persona = idPersona;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }
}
```

```
public void setApellido(String apellido) {
    this.apellido = apellido;
}

@Override
public String toString() {
    return "Persona{" + "id_persona=" + id_persona + ",
    nombre=" + nombre + ", apellido=" + apellido + '}';
}
}
```

VA CON JDBC

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo Conexion.java:

Dar click para ir al código

```
package datos;
import java.sql.*;

public class Conexion {
    private static String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static String JDBC_URL = "jdbc:mysql://localhost/sga?useSSL=false";
    private static String JDBC_USER = "root";
    private static String JDBC_PASS = "admin";
    private static Driver driver = null;

    public static synchronized Connection getConnection()
        throws SQLException {
        if (driver == null) {
            try {
                Class jdbcDriverClass = Class.forName(JDBC_DRIVER);
                driver = (Driver) jdbcDriverClass.newInstance();
                DriverManager.registerDriver(driver);
            } catch (Exception e) {
                System.out.println("Fallo en cargar el driver JDBC");
                e.printStackTrace();
            }
        }
        return DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASS);
    }
}
```

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

PASO 6. MODIFICAMOS EL CÓDIGO (CONT)

Archivo Conexion.java:

Dar click para ir al código

```
public static void close(ResultSet rs) {  
    try {  
        if (rs != null) {  
            rs.close();  
        }  
    } catch (SQLException sqle) {  
        sqle.printStackTrace();  
    }  
}
```

```
public static void close(PreparedStatement stmt) {  
    try {  
        if (stmt != null) {  
            stmt.close();  
        }  
    } catch (SQLException sqle) {  
        sqle.printStackTrace();  
    }  
}
```

//Cierre de la conexion

```
public static void close(Connection conn) {  
    try {  
        if (conn != null) {  
            conn.close();  
        }  
    } catch (SQLException sqle) {  
        sqle.printStackTrace();  
    }  
}
```

PASO 7. MODIFICAMOS EL CÓDIGO

Archivo PersonasJDBC.java:

Dar click para ir al código

```
package datos;

import domain.Persona;
import java.sql.*;
import java.util.*;

public class PersonasJDBC {
    //Nos apoyamos de la llave primaria autoincrementable de MySql
    //por lo que se omite el campo de persona_id
    //Se utiliza un preparedStatement, por lo que podemos
    //utilizar parametros (signos de ?)
    //los cuales posteriormente será sustituidos por el parametro respectivo

    private final String SQL_INSERT
        = "INSERT INTO persona(nombre, apellido) VALUES(?,?)";

    private final String SQL_UPDATE
        = "UPDATE persona SET nombre=?, apellido=? WHERE id_persona=?";

    private final String SQL_DELETE
        = "DELETE FROM persona WHERE id_persona = ?";

    private final String SQL_SELECT
        = "SELECT id_persona, nombre, apellido FROM persona ORDER BY id_persona";
}
```

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

PASO 7. MODIFICAMOS EL CÓDIGO (CONT)

Archivo PersonasJDBC.java:

Dar click para ir al código

```
public int insert(String nombre, String apellido) {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null; //no se utiliza en este ejercicio

    int rows = 0; //registros afectados
    try {
        conn = Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_INSERT);
        int index = 1; //contador de columnas
        stmt.setString(index++, nombre); //param 1 => ?
        stmt.setString(index++, apellido); //param 2 => ?
        System.out.println("Ejecutando query:" + SQL_INSERT);
        rows = stmt.executeUpdate(); //no. registros afectados
        System.out.println("Registros afectados:" + rows);

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(stmt);
        Conexion.close(conn);
    }

    return rows;
}
```

```
public int update(int id_persona, String nombre, String apellido)
{
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = Conexion.getConnection();
        System.out.println("Ejecutando query:" + SQL_UPDATE);
        stmt = conn.prepareStatement(SQL_UPDATE);
        int index = 1;
        stmt.setString(index++, nombre);
        stmt.setString(index++, apellido);
        stmt.setInt(index, id_persona);
        rows = stmt.executeUpdate();
        System.out.println("Registros actualizados:" + rows);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(stmt);
        Conexion.close(conn);
    }

    return rows;
}
```


PASO 7. MODIFICAMOS EL CÓDIGO (CONT)

Archivo PersonasJDBC.java:

Dar click para ir al código

```
public int delete(int id_persona) {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = Conexion.getConnection();
        System.out.println("Ejecutando query:" + SQL_DELETE);
        stmt = conn.prepareStatement(SQL_DELETE);
        stmt.setInt(1, id_persona);
        rows = stmt.executeUpdate();
        System.out.println("Registros eliminados:" + rows);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(stmt);
        Conexion.close(conn);
    }
    return rows;
}
```

```
public List<Persona> select() {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    Persona persona = null;
    List<Persona> personas = new ArrayList<Persona>();
    try {
        conn = Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_SELECT);
        rs = stmt.executeQuery();
        while (rs.next()) {
            int id_persona = rs.getInt(1);
            String nombre = rs.getString(2);
            String apellido = rs.getString(3);
            persona = new Persona();
            persona.setId_persona(id_persona);
            persona.setNombre(nombre);
            persona.setApellido(apellido);
            personas.add(persona);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        Conexion.close(rs);
        Conexion.close(stmt);
        Conexion.close(conn);
    }
    return personas;
}
```

PASO 8. MODIFICAMOS EL CÓDIGO (CONT)

Archivo ManejoPersonas.java:

Dar click para ir al código

```
package manejopersonas;

import datos.PersonasJDBC;
import domain.Persona;
import java.util.List;

public class ManejoPersonas {

    public static void main(String[] args) {
        PersonasJDBC personasJDBC = new PersonasJDBC();
        //Prueba del metodo insert
        //personasJDBC.insert("Alberto", "Juarez");

        //Prueba del metodo update
        //personasJDBC.update(2, "Nombre3", "Apellido3");

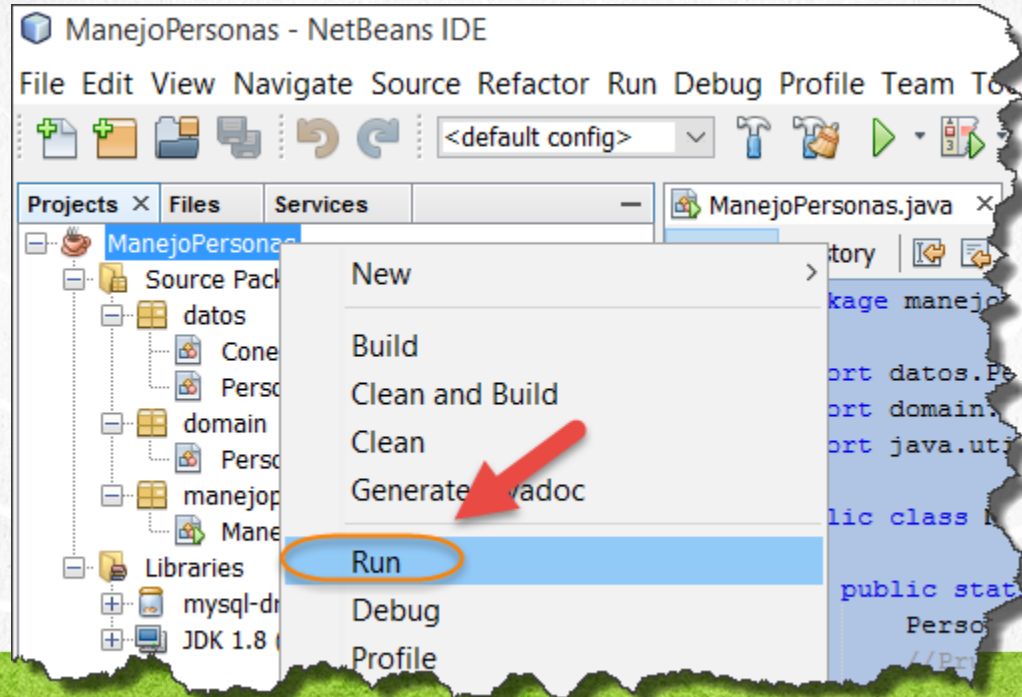
        //Prueba del metodo delete
        //personasJDBC.delete(1);

        //Prueba del metodo select
        //Uso de un objeto persona para encapsular la informacion
        //de un registro de base de datos
        List<Persona> personas = personasJDBC.select();
        for (Persona persona : personas) {
            System.out.print(persona);
            System.out.println("");
        }
    }
}
```

Ir ejecutando cada caso
e ir comprobando que
funcione correctamente
en la base de datos o
haciendo un select

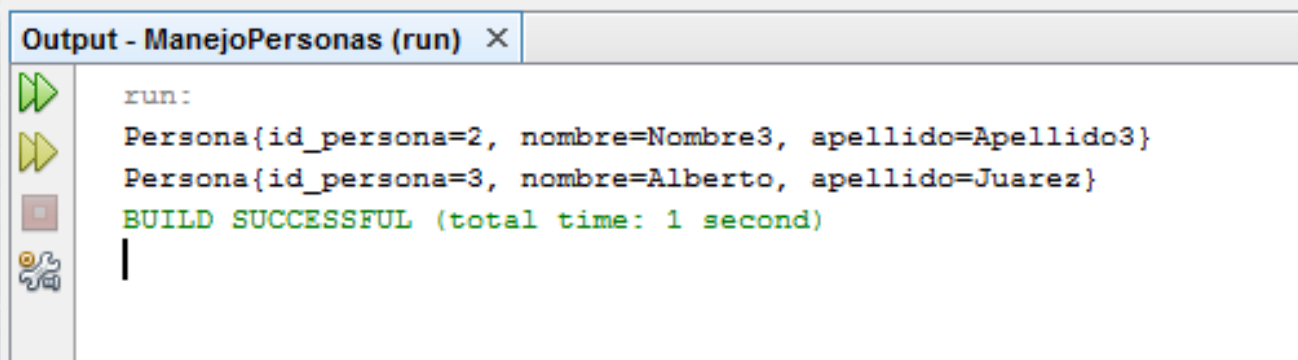
PASO 9. EJECUTAMOS EL PROYECTO

Ejecutamos nuestro proyecto. Damos click derecho -> Run:



PASO 9. EJECUTAMOS EL PROYECTO (CONT)

El resultado es como sigue:

A screenshot of an IDE's output window titled "Output - ManejoPersonas (run) x". The window contains the following text: "run:", "Persona{id_persona=2, nombre=Nombre3, apellido=Apellido3}", "Persona{id_persona=3, nombre=Alberto, apellido=Juarez}", "BUILD SUCCESSFUL (total time: 1 second)", and a cursor on the next line. On the left side of the window, there are four icons: a green play button, a yellow play button, a red stop button, and a bug icon.

```
run:
Persona{id_persona=2, nombre=Nombre3, apellido=Apellido3}
Persona{id_persona=3, nombre=Alberto, apellido=Juarez}
BUILD SUCCESSFUL (total time: 1 second)
|
```


CONCLUSIÓN DEL EJERCICIO

- Con este ejercicio hemos puesto en práctica varios conceptos, como la separación de responsabilidades básica en una clase de conexión, otra clase de tipo Persona que representa un registro en la base de datos, y otra clase PersonaJDBC que contiene las operaciones básicas para manipular esta información en la tabla personas de la base de datos, tales como insertar, actualizar, eliminar y seleccionar.



Experiencia y Conocimiento para tu vida

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

CURSO ONLINE

JAVA CON JDBC

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx